**PL/SQL TASKS**

**DONE BY,**

**P.DILIPKUMAR**

**Task 1 (Create Table, Insert Data to Table, Write Queries)**

Create the below table.

Policy Number Varchar2(20)

Plan Code Varchar2(10)

Age Number

Sum Assured Number

Premium Number

Start Date Date

End Date Date

Insert below data to the table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Policy Number** | **Plan Code** | **Age** | **Sum Assured** | **Premium** | **Start Date** | **End Date** |
| POL\_1 | P1 | 30 | 50000 | 500 | 01/01/2020 | 31/12/2020 |
| POL\_2 | P1 | 32 | 80000 | 800 | 05-Jan-2020 | 04-Jan-2021 |
| POL\_3 | P2 | 40 | 100000 | 100 | 01-02-2020 | 31-01-2020 |
| POL\_4 | P1 | 38 | 250000 | 2500 | 15-02-2020 | 14-02-2020 |
| PL\_5 | P2 | 41 | 700000 | 700 | 21st Jan 2020 | 20th Jan 2021 |

**Create a table Insurance**

DROP TABLE INSURANCE;

CREATE TABLE INSURANCE (

POLICY\_NUMBER VARCHAR2(10),

PLAN\_CODE VARCHAR2(10),

AGE NUMBER,

SUM\_ASSURED NUMBER,

PREMIUM NUMBER,

START\_DATE DATE,

END\_DATE DATE

);

**Insert record into a insurance table**

INSERT INTO INSURANCE(POLICY\_NUMBER,PLAN\_CODE,AGE,SUM\_ASSURED,PREMIUM,START\_DATE,END\_DATE) VALUES('POL\_1' ,'P1' ,30 ,50000, 500, '01-JAN-2020', '31-DEC-2020');

INSERT INTO INSURANCE(POLICY\_NUMBER,PLAN\_CODE,AGE,SUM\_ASSURED,PREMIUM,START\_DATE,END\_DATE) VALUES('POL\_2' ,'P1' ,32 ,80000 ,800 ,'05-JAN-2020', '04-JAN-2021');

INSERT INTO INSURANCE(POLICY\_NUMBER,PLAN\_CODE,AGE,SUM\_ASSURED,PREMIUM,START\_DATE,END\_DATE) VALUES('POL\_3' ,'P2' ,40 ,100000 ,100 ,'01-FEB-2020', '31-JAN-2020');

INSERT INTO INSURANCE(POLICY\_NUMBER,PLAN\_CODE,AGE,SUM\_ASSURED,PREMIUM,START\_DATE,END\_DATE) VALUES('POL\_4' ,'P1',38 ,250000 ,2500 ,'15-FEB-2020' ,'14-FEB-2021');

INSERT INTO INSURANCE(POLICY\_NUMBER,PLAN\_CODE,AGE,SUM\_ASSURED,PREMIUM,START\_DATE,END\_DATE) VALUES('PL\_5',' P2', 41, 700000, 700 ,'21-JAN- 2020', '20-JAN-2021');

SELECT \* FROM INSURANCE;

ALTER TABLE INSURANCE ADD CONSTRAINTS INSURANCE\_PK PRIMARY KEY(POLICY\_NUMBER);

DESC INSURANCE;

**Write Queries for the below:-**

|  |  |
| --- | --- |
| **1** | Plan Code Wise Policy Count |
| 2 | Plan Code Wise Total Premium, Total Sum Assured |
| 3 | Find policies with age between 30 and 40 |
| 4 | Find Policies with policy number starting with PL |
| 5 | Select Policies age wise in descending order |
| 6 | Plan Code wise count of policies where age greater than 30 |
| 7 | Find policies with start date in January |
| 8 | Select policies premium wise ascending, sum assured wise descending |
| 9 | Plan Code wise Total Premium greater than 1000 |
| 10 | Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000 |
|  |  |

# /\*QUERIES\*/

1. **Plan Code Wise Policy Count**

**SQL Code:**

SELECT PLAN\_CODE,COUNT(POLICY\_NUMBER) FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

**Plan Code Wise Policy Count using PL/SQL Cursors**

DECLARE

v\_plancode INSURANCE.PLAN\_CODE%TYPE;

v\_count NUMBER;

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE,COUNT(\*)

FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

OPEN v\_insur\_cur;

LOOP

FETCH v\_insur\_cur INTO v\_plancode, v\_count;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||v\_plancode);

DBMS\_OUTPUT.PUT\_LINE('POLICY COUNT: '||v\_count);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur;

END;

**Plan Code Wise Policy Count using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE,COUNT(\*) AS "POLICY\_COUNT"

FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

FOR insurance\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||insurance\_rec.PLAN\_CODE);

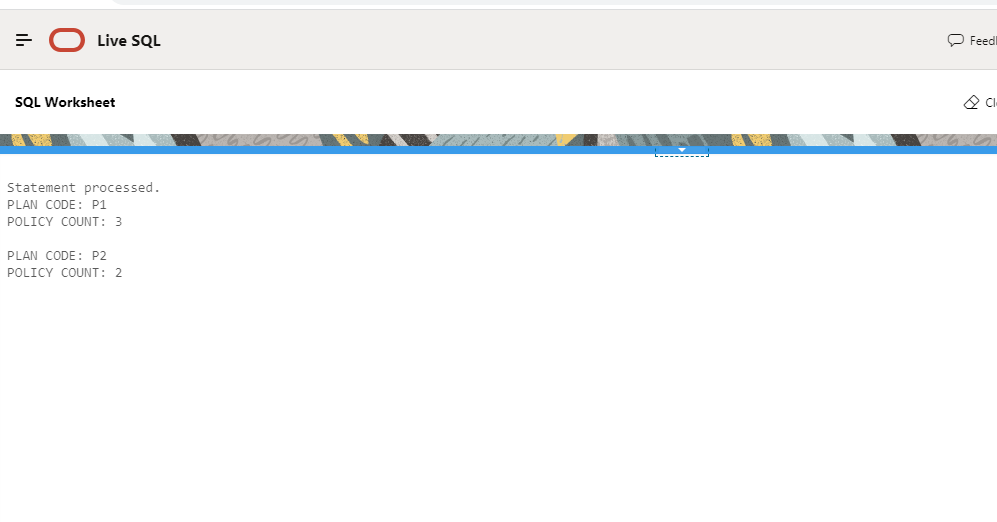
DBMS\_OUTPUT.PUT\_LINE('POLICY COUNT: '||insurance\_rec.POLICY\_COUNT);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Plan Code Wise Total Premium, Total Sum Assured**

**SQL Code:**

SELECT PLAN\_CODE,SUM(PREMIUM),SUM(SUM\_ASSURED)

FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

**Plan Code Wise Total Premium, Total Sum Assured using PL/SQL Cursor**

DECLARE

v\_plancode INSURANCE.PLAN\_CODE%TYPE;

v\_totalsumassured NUMBER;

v\_totalpremium NUMBER;

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE,SUM(PREMIUM),SUM(SUM\_ASSURED)

FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

OPEN v\_insur\_cur;

LOOP

FETCH v\_insur\_cur INTO v\_plancode,v\_totalpremium,v\_totalsumassured;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||v\_plancode);

DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM : '||v\_totalpremium);

DBMS\_OUTPUT.PUT\_LINE('TOTAL SUM ASSURED : '||v\_totalsumassured);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur;

END;

**Plan Code Wise Total Premium, Total Sum Assured using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE,SUM(PREMIUM) AS "TOTAL\_PREMIUM",SUM(SUM\_ASSURED) AS "TOTAL\_ASSUERD"

FROM INSURANCE

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

FOR insur\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||insur\_rec.PLAN\_CODE);

DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM: '||insur\_rec.TOTAL\_PREMIUM);

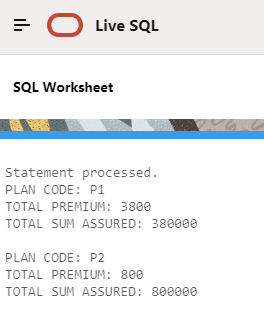
DBMS\_OUTPUT.PUT\_LINE('TOTAL SUM ASSURED: '||insur\_rec.TOTAL\_ASSUERD);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Find policies with age between 30 and 40**

**SQL code:**

SELECT POLICY\_NUMBER, AGE

FROM INSURANCE

WHERE AGE BETWEEN 30 AND 40;

**Find policies with age between 30 and 40 using PL/SQL Cursor**

DECLARE

v\_policynumber INSURANCE.POLICY\_NUMBER%TYPE;

v\_age INSURANCE.AGE%TYPE;

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER, AGE

FROM INSURANCE

WHERE AGE BETWEEN 30 AND 40;

BEGIN

OPEN v\_insur\_cur ;

LOOP

FETCH v\_insur\_cur INTO v\_policynumber,v\_age;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||v\_policynumber);

DBMS\_OUTPUT.PUT\_LINE('AGE : '||v\_age);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur ;

END;

**Find policies with age between 30 and 40 using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER, AGE

FROM INSURANCE

WHERE AGE BETWEEN 30 AND 40;

BEGIN

FOR insur\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||insur\_rec.POLICY\_NUMBER);

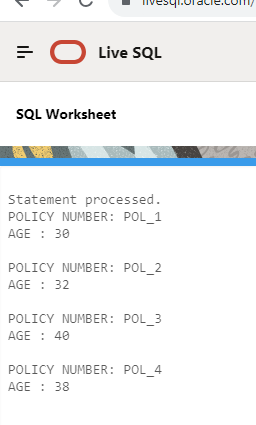
DBMS\_OUTPUT.PUT\_LINE('AGE : '||insur\_rec.AGE);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Find Policies with policy number starting with PL**

**SQL Code:**

SELECT POLICY\_NUMBER FROM INSURANCE

WHERE POLICY\_NUMBER LIKE'PL%';

**Find Policies with policy number starting with PL using PL/SQL Cursor**

DECLARE

v\_policynumber INSURANCE.POLICY\_NUMBER%TYPE;

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER FROM INSURANCE

WHERE POLICY\_NUMBER LIKE'PL%';

BEGIN

OPEN v\_insur\_cur ;

LOOP

FETCH v\_insur\_cur INTO v\_policynumber;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||v\_policynumber);

END LOOP;

CLOSE v\_insur\_cur ;

END;

**Find Policies with policy number starting with PL using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER

FROM INSURANCE

WHERE POLICY\_NUMBER LIKE '%PL%';

BEGIN

FOR insur\_rec IN v\_insur\_cur

LOOP

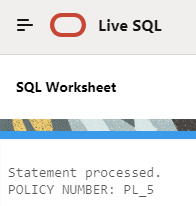
EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||insur\_rec.POLICY\_NUMBER);

END LOOP;

END;

**Result:**



1. **Select Policies age wise in descending order**

**SQL code:**

SELECT POLICY\_NUMBER,AGE

FROM INSURANCE

ORDER BY AGE DESC;

**Select Policies age wise in descending order using PL/SQL Cursor**

DECLARE

v\_policynumber INSURANCE.POLICY\_NUMBER%TYPE;

v\_age INSURANCE.AGE%TYPE;

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER,AGE

FROM INSURANCE

ORDER BY AGE DESC;

BEGIN

OPEN v\_insur\_cur;

LOOP

FETCH v\_insur\_cur INTO v\_policynumber,v\_age;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||v\_policynumber);

DBMS\_OUTPUT.PUT\_LINE('AGE : '||v\_age);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur;

END;

**Select Policies age wise in descending order using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER,AGE

FROM INSURANCE

ORDER BY AGE DESC;

BEGIN

FOR ins\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||ins\_rec.POLICY\_NUMBER);

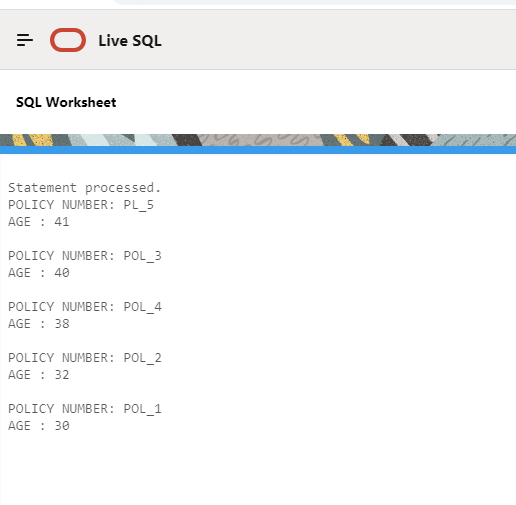
DBMS\_OUTPUT.PUT\_LINE('AGE : '||ins\_rec.AGE);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Plan Code wise count of policies where age greater than 30**

**SQL Code:**

SELECT PLAN\_CODE, COUNT (POLICY\_NUMBER) AS POLICY\_COUNT FROM INSURANCE

WHERE AGE>30

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

**Plan Code wise count of policies where age greater than 30 using PL/SQL Cursor**

DECLARE

v\_plancode INSURANCE.PLAN\_CODE%TYPE;

v\_count NUMBER;

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE, COUNT (POLICY\_NUMBER)

FROM INSURANCE

WHERE AGE>30

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

OPEN v\_insur\_cur;

LOOP

FETCH v\_insur\_cur INTO v\_plancode,v\_count;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE : '||v\_plancode);

DBMS\_OUTPUT.PUT\_LINE(COUNT : '||v\_count);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur;

END;

**Plan Code wise count of policies where age greater than 30 using PL/SQL Cursor For loop**

DECLARE

CURSOR v\_insur\_cur IS SELECT PLAN\_CODE, COUNT (POLICY\_NUMBER) AS "POLICY\_COUNT"

FROM INSURANCE

WHERE AGE>30

GROUP BY PLAN\_CODE

ORDER BY PLAN\_CODE;

BEGIN

FOR ins\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||ins\_rec.PLAN\_CODE);

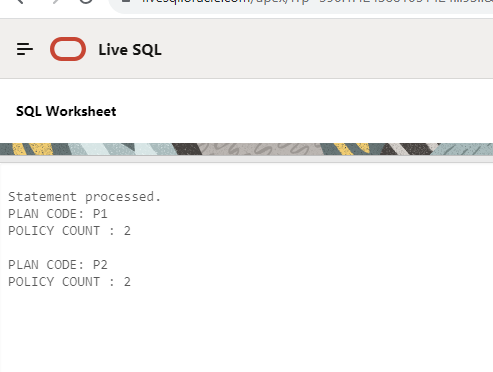
DBMS\_OUTPUT.PUT\_LINE('POLICY COUNT : '||ins\_rec.POLICY\_COUNT);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Find policies with start date in January**

**SQL code:**

SELECT POLICY\_NUMBER, START\_DATE FROM INSURANCE

WHERE START\_DATE LIKE '%JAN%';

**Find policies with start date in January using PL/SQL cursor**

DECLARE

v\_policynumber INSURANCE.POLICY\_NUMBER%TYPE;

v\_startdate INSURANCE.START\_DATE%TYPE;

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER, START\_DATE FROM INSURANCE

WHERE START\_DATE LIKE '%JAN%';

BEGIN

OPEN v\_insur\_cur;

LOOP

FETCH v\_insur\_cur INTO v\_policynumber,v\_startdate;

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||v\_policynumber);

DBMS\_OUTPUT.PUT\_LINE('START DATE: '||v\_startdate);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_insur\_cur;

END;

**Find policies with start date in January using PL/SQL cursor**

DECLARE

CURSOR v\_insur\_cur IS SELECT POLICY\_NUMBER, START\_DATE FROM INSURANCE

WHERE START\_DATE LIKE '%JAN%';

BEGIN

FOR ins\_rec IN v\_insur\_cur

LOOP

EXIT WHEN v\_insur\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER: '||ins\_rec.POLICY\_NUMBER);

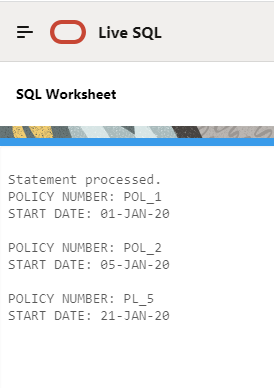
DBMS\_OUTPUT.PUT\_LINE('START DATE: '||ins\_rec.START\_DATE);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Select policies premium wise ascending, sum assured wise descending**

**SQL code:**

SELECT POLICY\_NUMBER, PREMIUM, SUM\_ASSURED FROM INSURANCE

ORDER BY PREMIUM, SUM\_ASSURED DESC;

**Select policies premium wise ascending, sum assured wise descending using PL/SQL cursor**

DECLARE

v\_policynumber INSURANCE.POLICY\_NUMBER%TYPE;

v\_premium INSURANCE.PREMIUM %TYPE;

v\_summassured INSURANCE.SUM\_ASSURED%TYPE;

CURSOR v\_ins\_cur IS SELECT POLICY\_NUMBER,PREMIUM,SUM\_ASSURED

FROM INSURANCE

ORDER BY PREMIUM ASC,

SUM\_ASSURED DESC;

BEGIN

OPEN v\_ins\_cur;

LOOP

FETCH v\_ins\_cur INTO v\_policynumber,v\_premium,v\_summassured ;

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER : '||v\_policynumber);

DBMS\_OUTPUT.PUT\_LINE('PREMIUM : '||v\_premium);

DBMS\_OUTPUT.PUT\_LINE('SUM ASSURED: '||v\_summassured);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_ins\_cur;

END;

**Select policies premium wise ascending, sum assured wise descending using PL/SQL cursor for loop**

DECLARE

CURSOR v\_ins\_cur IS SELECT POLICY\_NUMBER,PREMIUM,SUM\_ASSURED

FROM INSURANCE

ORDER BY PREMIUM ASC,

SUM\_ASSURED DESC;

BEGIN

FOR ins\_rec IN v\_ins\_cur

LOOP

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('POLICY NUMBER : '||ins\_rec.POLICY\_NUMBER);

DBMS\_OUTPUT.PUT\_LINE('PREMIUM : '||ins\_rec.PREMIUM);

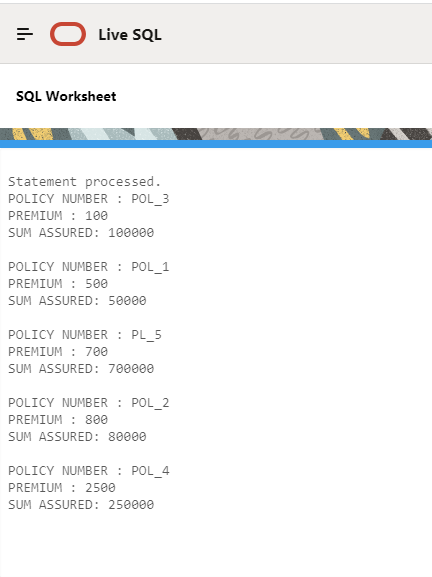
DBMS\_OUTPUT.PUT\_LINE('SUM ASSURED: '||ins\_rec.SUM\_ASSURED);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Plan Code wise Total Premium greater than 1000**

**SQL code:**

SELECT PLAN\_CODE, SUM (PREMIUM) AS TOTAL\_PREMIUM FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM (PREMIUM)>1000

ORDER BY PLAN\_CODE;

**Plan Code wise Total Premium greater than 1000 using PL/SQL cursor**

DECLARE

v\_plancode INSURANCE.PLAN\_CODE%TYPE;

v\_totalprem NUMBER;

CURSOR v\_ins\_cur IS SELECT PLAN\_CODE, SUM (PREMIUM)

FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM (PREMIUM)>1000

ORDER BY PLAN\_CODE;

BEGIN

OPEN v\_ins\_cur;

LOOP

FETCH v\_ins\_cur INTO v\_plancode,v\_totalprem;

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||v\_plancode);

DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM : '||v\_totalprem);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

CLOSE v\_ins\_cur;

END;

**Plan Code wise Total Premium greater than 1000 using PL/SQL cursor for loop**

DECLARE

CURSOR v\_ins\_cur IS SELECT PLAN\_CODE, SUM (PREMIUM) AS "TOTAL\_PREMIUM"

FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM (PREMIUM)>1000

ORDER BY PLAN\_CODE;

BEGIN

FOR ins\_rec IN v\_ins\_cur

LOOP

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||ins\_rec.PLAN\_CODE);

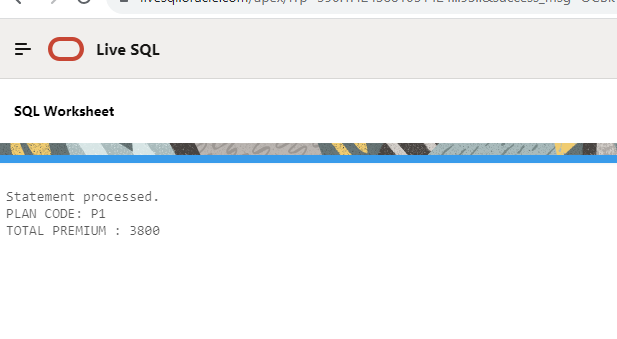
DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM : '||ins\_rec.TOTAL\_PREMIUM);

DBMS\_OUTPUT.PUT\_LINE(' ');

END LOOP;

END;

**Result:**



1. **Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000**

**SQL code:**

SELECT PLAN\_CODE,SUM(PREMIUM),SUM(SUM\_ASSURED)

FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM(PREMIUM)<3000 AND SUM(SUM\_ASSURED)>50000

ORDER BY PLAN\_CODE;

**Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000 using PL/SQL cursor**

DECLARE

v\_plancode INSURANCE.PLAN\_CODE%TYPE;

v\_totalsumassured NUMBER;

v\_totalprem NUMBER;

CURSOR v\_ins\_cur IS SELECT PLAN\_CODE,SUM(PREMIUM),SUM(SUM\_ASSURED)

FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM(PREMIUM)<3000 AND SUM(SUM\_ASSURED)>50000

ORDER BY PLAN\_CODE;

BEGIN

OPEN v\_ins\_cur ;

LOOP

FETCH v\_ins\_cur INTO v\_plancode,v\_totalsumassured,v\_totalprem;

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||v\_plancode);

DBMS\_OUTPUT.PUT\_LINE('TOTAL SUM ASSURED : '||v\_totalsumassured);

DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM : '||v\_totalprem);

END LOOP;

CLOSE v\_ins\_cur ;

END;

**Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000 using PL/SQL cursor for loop**

DECLARE

CURSOR v\_ins\_cur IS SELECT PLAN\_CODE,SUM(PREMIUM) AS "TOT\_PREM",SUM(SUM\_ASSURED) AS "TOT\_SUM\_ASSURED"

FROM INSURANCE

GROUP BY PLAN\_CODE

HAVING SUM(PREMIUM)<3000 AND SUM(SUM\_ASSURED)>50000

ORDER BY PLAN\_CODE;

BEGIN

FOR ins\_rec IN v\_ins\_cur

LOOP

EXIT WHEN v\_ins\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PLAN CODE: '||ins\_rec.PLAN\_CODE);

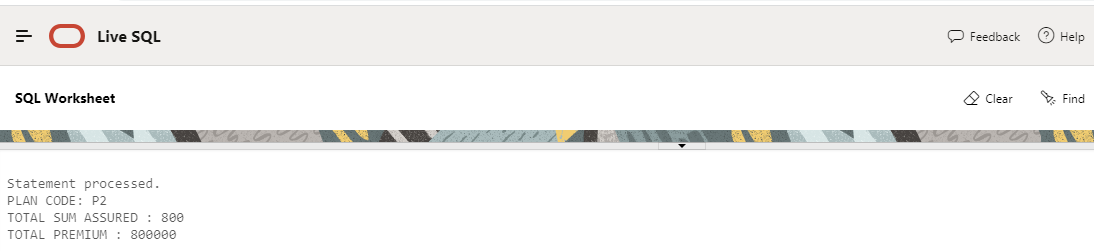
DBMS\_OUTPUT.PUT\_LINE('TOTAL SUM ASSURED : '||ins\_rec.TOT\_SUM\_ASSURED);

DBMS\_OUTPUT.PUT\_LINE('TOTAL PREMIUM : '||ins\_rec.TOT\_PREM);

END LOOP;

END;

**Result:**



**Task 2 (Create Tables, Insert Data to Table 1, Write Procedure to Populate Table 2)**

Create 2 tables Class Grade and Student Grade

Table - Class Grade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Mark From** | **Mark To** | | **Grade** |
| 1 | 1 | 39 | | Fail |
| 1 | 40 | 49 | | 3rd Class |
| 1 | 50 | 59 | | 2nd Class |
| 1 | 60 | 79 | | 1st Class |
| 1 | 80 | | 100 | Distinction |
|  |  |  | |  |
| Insert above date into Class Grade table. | | | |  |
| Table – Student Grade | |  | |  |
| **Student** | **Class** | **Mark** | | **Grade** |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |

|  |  |
| --- | --- |
| Write a procedure with input (Student, Class and Mark) | |
| Find the Grade from Class Grade table based on Class & Mark |
| Insert Input data and grade into Student Grade Table |

**PL/SQL code:**

**Create a table class\_grade**

DROP TABLE CLASS\_GRADE;

CREATE TABLE CLASS\_GRADE

(

CLASS NUMBER,

MARK\_FROM NUMBER,

MARK\_TO NUMBER,

GRADE VARCHAR2(50)

);

**Insert values into the class\_grade table**

INSERT INTO CLASS\_GRADE(CLASS,MARK\_FROM,MARK\_TO,GRADE) VALUES (1,1,39,'FAIL');

INSERT INTO CLASS\_GRADE(CLASS,MARK\_FROM,MARK\_TO,GRADE) VALUES (1,40,49,'3RD CLASS');

INSERT INTO CLASS\_GRADE(CLASS,MARK\_FROM,MARK\_TO,GRADE) VALUES (1,50,59,'2ND CLASS');

INSERT INTO CLASS\_GRADE(CLASS,MARK\_FROM,MARK\_TO,GRADE) VALUES (1,60,79,'1ST CLASS');

INSERT INTO CLASS\_GRADE(CLASS,MARK\_FROM,MARK\_TO,GRADE) VALUES (1,80,100,'DISTINCTION');

SELECT \* FROM CLASS\_GRADE;

DROP TABLE STUDENT\_GRADE;

**Create a table student\_grade**

CREATE TABLE STUDENT\_GRADE

(

STUDENT NUMBER,

CLASS NUMBER,

MARK NUMBER,

GRADE VARCHAR2(50)

);

SELECT \* FROM STUDENT\_GRADE;

**Create a procedure for finding grade result**

CREATE OR REPLACE PROCEDURE student\_mark\_procedure(v\_student NUMBER,

v\_class NUMBER,

v\_mark NUMBER

)

AS

v\_graderesult VARCHAR2(50);

BEGIN

SELECT GRADE INTO v\_graderesult

FROM CLASS\_GRADE

WHERE CLASS = v\_class

AND MARK\_FROM <= v\_mark

AND MARK\_TO >= v\_mark;

INSERT INTO STUDENT\_GRADE (STUDENT,CLASS, MARK,GRADE) VALUES (v\_student,v\_class,v\_mark,v\_graderesult) ;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found, please check to the class grade table');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An Error occured...'||SQLERRM);

END student\_mark\_procedure;

**Execute a procedure**

BEGIN

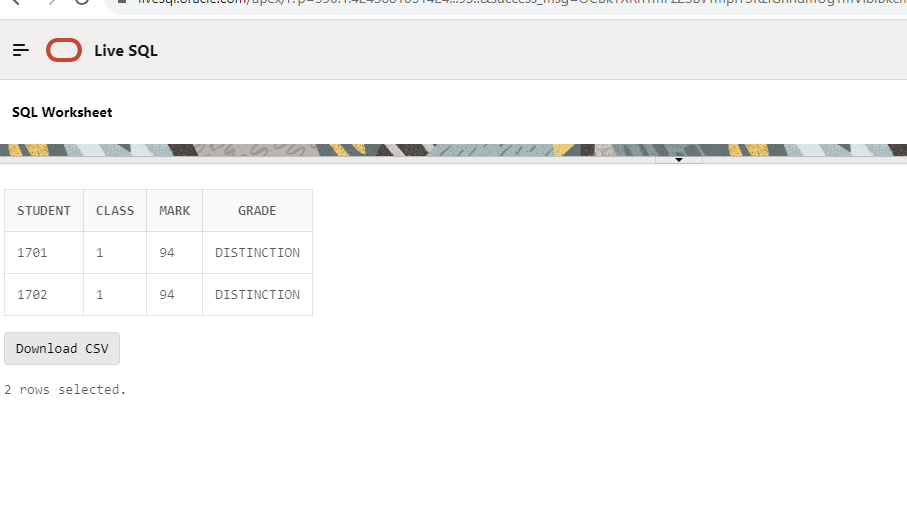
student\_mark\_procedure(1701,1,94);

END;

EXEC student\_mark\_procedure(1702,1,94);

EXEC student\_mark\_procedure(1703,1,-5);

**Result:**



**Task 3 (Create Tables, Insert Data to Table 1, Write Procedure to Populate Table 2)**

Create 2 tables inventory master and inventory purchase

Table - Tbl\_inventory\_master

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Item Code** | **Item Name** | **Stock** | **Selling Rate** | **Buying Rate** |  |  |  |  |  |  |
| IT01 | Pen | 0 | 10 | 6 |  |  |  |  |  |  |
| IT02 | Pencil | 0 | 5 | 3 |  |  |  |  |  |  |
| IT03 | Eraser | 0 | 3 | 1 |  |  |  |  |  |  |
| IT04 | Scale | 0 | 25 | 15 |  |  |  |  |  |  |
| Create DML Statements to insert the above data to inventory master table.  Table - Tbl\_inventory\_purchase | | | | | | | | | |  |
| **Item code** | **Purchase date** | **Purchase Qty** | **Selling Rate** | **Purchase Amt** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Write a Procedure to insert data into inventory purchase table by passing Item Code, Purchase Date and Purchase quantity as parameters.

Before inserting into above table find the selling rate of that item code from tbl\_inventory\_master and calculate purchase amount (rate \* purchase qty)

While Inserting into the above table, insert input data along with selling rate and calculated purchase amount

**Create a table Inventory\_master**

DROP TABLE INVENTORY\_MASTER;

CREATE TABLE INVENTORY\_MASTER

(

ITEM\_CODE VARCHAR2(50),

ITEM\_NAME VARCHAR2(200),

STOCK NUMBER,

SELLING\_RATE NUMBER,

BUYING\_RATE NUMBER

);

SELECT \* FROM INVENTORY\_MASTER;

**Insert values into the table inventory\_master**

INSERT INTO INVENTORY\_MASTER(ITEM\_CODE,ITEM\_NAME ,STOCK ,SELLING\_RATE ,BUYING\_RATE ) VALUES ( 'IT01','Pen',0,10,6);

INSERT INTO INVENTORY\_MASTER(ITEM\_CODE,ITEM\_NAME ,STOCK ,SELLING\_RATE ,BUYING\_RATE ) VALUES ( 'IT02','PENCIL',0,5,3);

INSERT INTO INVENTORY\_MASTER(ITEM\_CODE,ITEM\_NAME ,STOCK ,SELLING\_RATE ,BUYING\_RATE ) VALUES ( 'IT04','ERASER',0,3,1);

INSERT INTO INVENTORY\_MASTER(ITEM\_CODE,ITEM\_NAME ,STOCK ,SELLING\_RATE ,BUYING\_RATE ) VALUES ( 'IT04','SCLAE',0,25,15);

**Create a table Inventory\_purchase**

DROP TABLE INVENTORY\_PURCHASE;

CREATE TABLE INVENTORY\_PURCHASE

(

ITEM\_CODE VARCHAR2(50),

PURCHASE\_DATE DATE,

PURCHASE\_QTY NUMBER,

SELLING\_RATE NUMBER,

PURCHASE\_AMT NUMBER

);

**Create a procedure to insert inventery\_purchase table**

DROP PROCEDURE insert\_inventory;

CREATE OR REPLACE PROCEDURE insert\_inventory

(

v\_itemcode INVENTORY\_MASTER.ITEM\_CODE%TYPE,

v\_purchase\_date DATE,

v\_purchase\_qty NUMBER

)

AS

v\_selling\_rate NUMBER;

v\_purchase\_amt NUMBER;

BEGIN

SELECT SELLING\_RATE

INTO v\_selling\_rate

FROM INVENTORY\_MASTER

WHERE ITEM\_CODE = v\_itemcode;

v\_purchase\_amt := v\_selling\_rate \* v\_purchase\_qty;

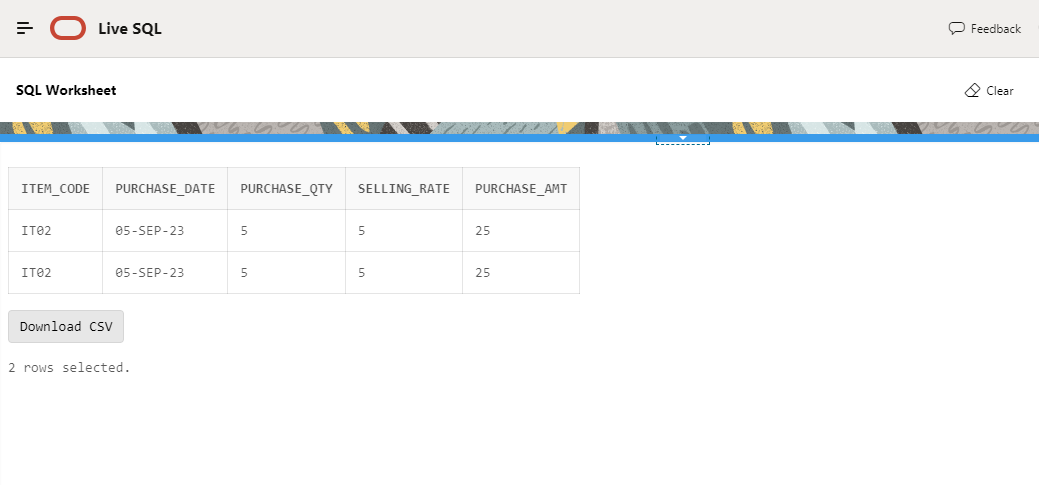
INSERT INTO INVENTORY\_PURCHASE (ITEM\_CODE,PURCHASE\_DATE,PURCHASE\_QTY,SELLING\_RATE,PURCHASE\_AMT) VALUES (v\_itemcode,v\_purchase\_date,v\_purchase\_qty,v\_selling\_rate,v\_purchase\_amt);

END insert\_inventory;

**Execute a procedure:**

EXECUTE insert\_inventory('IT02',SYSDATE,5);

**Result:**



**Task 4 (Create Tables, Write Procedure to Populate both the Tables)**

Create 2 tables card master and card details

**Card Master**

Card ID Varchar2(20)

Card Name Varchar2(100)

Prev Type Varchar2(1)

Prev Amount Number

Curr Type Varchar2(1)

Curr Amount Number

Net Type Varchar2(1)

Net Amount Number

Last\_trans\_Date Date

**Card Details**

Card ID Varchar2(20)

Type Varchar2(1)

Amount Number

Trans\_Date Date

Write a procedure with the following parameters

Card ID, Card Name, Type, Amount, Trans Date

If it is new card ID, a new record should be created in Card Master table.

Prev Type, Prev Amount should be Null.

Current Type, Current Amount should be Type and Amount from Input.

Net Type and Net amount should be Type and Amount from Input.

Last Trans Date should be Trans Date from Input.

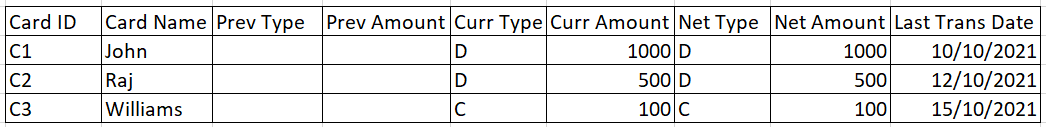
Card ID, Type, Amount, Trans date should be inserted into Card details table as it is from Input.

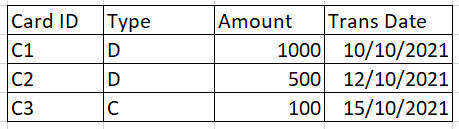
Exec Procedure P1 (C1, ‘John’, D, 1000, ‘10/10/2021’);

Exec Procedure P1 (C2, ‘Raj’, D, 500, ‘12/10/2021’);

Exec Procedure P1 (C3, ‘Williams ‘, C, 100, ‘15/10/2021’);

Above execution, should give the below results in card master and card details tables.





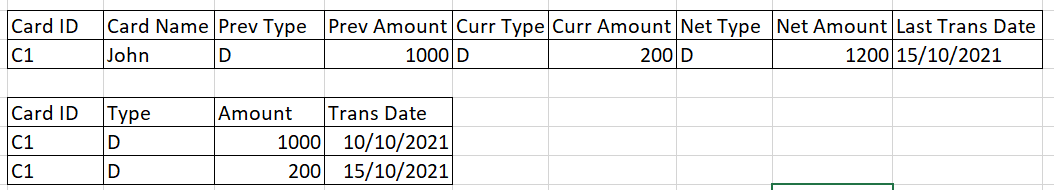
If the card already exists, in card master, data should be inserted as it is in card details table.

In Master existing data should be updated as below:-

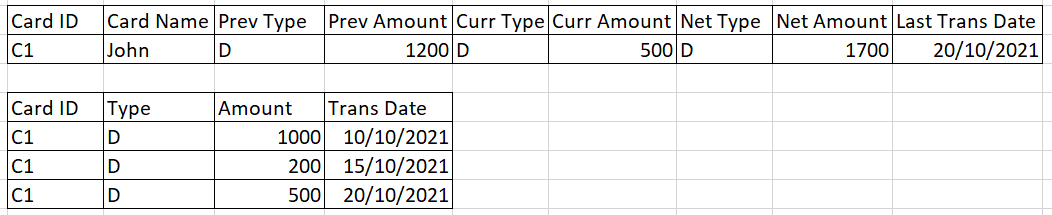
Prev Type, Prev Amount, Current Type, Current Amount and Net Type and Net amount should be calculated as the examples shown in the transactions below.

Last Trans Date should be Trans Date from Input.

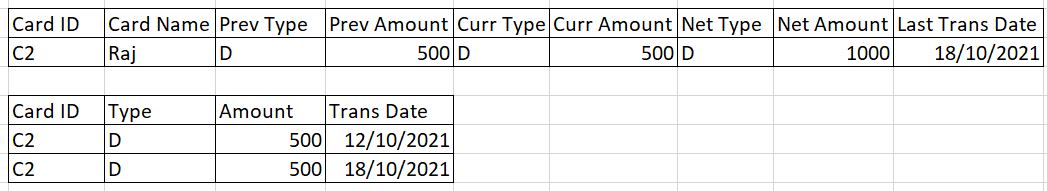
Exec Procedure P1 (C1, ‘John’, D, 200, ‘15/10/2021’);



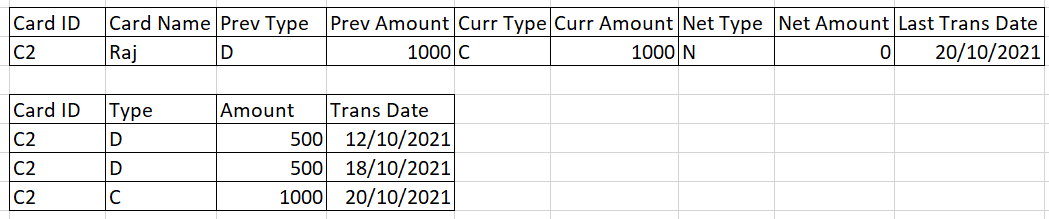
Exec Procedure P1 (C1, ‘John’, D, 500, ‘20/10/2021’);



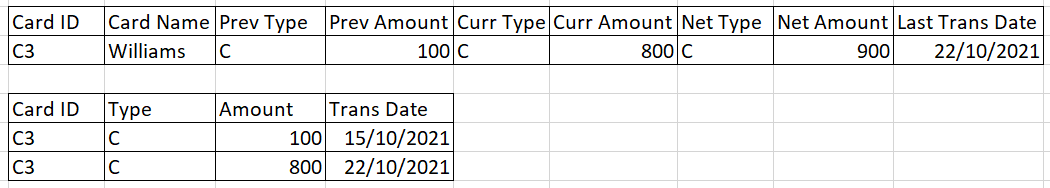
Exec Procedure P1 (C2, ‘Raj’, D, 500, ‘12/10/2021’);



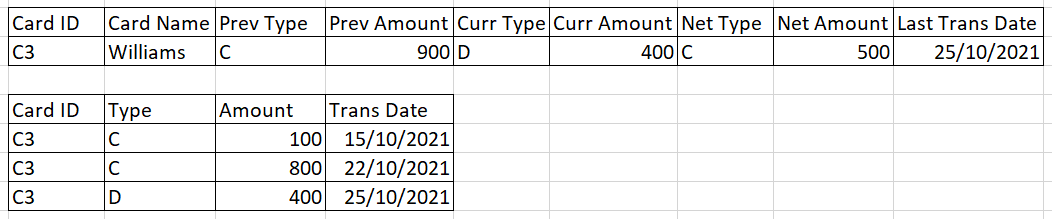
Exec Procedure P1 (C2, ‘Raj’, C, 1000, ‘20/10/2021’);



Exec Procedure P1 (C3, ‘Williams ‘, C, 800, ‘22/10/2021’);



Exec Procedure P1 (C3, ‘Williams ‘, D, 400, ‘25/10/2021’);



**Create a table card\_master**

DROP TABLE CARD\_MASTER;

CREATE TABLE CARD\_MASTER

(

CARD\_ID VARCHAR2(20),

CARD\_NAME VARCHAR2(100),

PREV\_TPYE VARCHAR2(1),

PREV\_AMOUNT NUMBER,

CURR\_TYPE VARCHAR2(1),

CURR\_AMOUNT NUMBER,

NET\_TYPE VARCHAR2(1),

NET\_AMOUNT NUMBER,

LAST\_TRANS\_DATE DATE

);

SELECT \* FROM CARD\_MASTER;

**Create a table card\_details**

DROP TABLE CARD\_DETAILS;

CREATE TABLE CARD\_DETAILS

(

CARD\_ID VARCHAR2(20),

TYPE VARCHAR2(1),

AMOUNT NUMBER,

TRANS\_DATE DATE

);

SELECT \* FROM CARD\_DETAILS;

**Create a procedure for both the tables:**

CREATE OR REPLACE PROCEDURE card\_proc (

P\_CARD\_ID VARCHAR2,

P\_CARD\_NAME VARCHAR2,

P\_TYPE VARCHAR2,

P\_AMOUNT NUMBER,

P\_TRANS\_DATE DATE

)

IS

V\_EXISTING\_CARD NUMBER;

V\_PREV\_TYPE VARCHAR2(1);

V\_PREV\_AMOUNT NUMBER;

V\_CURRENT\_TYPE VARCHAR2(1);

V\_CURRENT\_AMOUNT NUMBER;

V\_NET\_TYPE VARCHAR2(1);

V\_NET\_AMOUNT NUMBER;

BEGIN

**-- To Check if the card already exists in Card Master table**

SELECT COUNT(\*) INTO V\_EXISTING\_CARD FROM CARD\_MASTER WHERE CARD\_ID = P\_CARD\_ID;

IF V\_EXISTING\_CARD = 0 THEN

**-- Card is new, insert a new record into Card Master table**

INSERT INTO CARD\_MASTER (CARD\_ID, CARD\_NAME, PREV\_TPYE, PREV\_AMOUNT, CURR\_TYPE, CURR\_AMOUNT, NET\_TYPE, NET\_AMOUNT, LAST\_TRANS\_DATE)

VALUES (P\_CARD\_ID, P\_CARD\_NAME, NULL, NULL, P\_TYPE, P\_AMOUNT, P\_TYPE, P\_AMOUNT, P\_TRANS\_DATE);

ELSE

SELECT CURR\_TYPE, CURR\_AMOUNT INTO V\_PREV\_TYPE, V\_PREV\_AMOUNT

FROM CARD\_MASTER

WHERE CARD\_ID = P\_CARD\_ID;

V\_CURRENT\_TYPE := P\_TYPE;

V\_CURRENT\_AMOUNT := P\_AMOUNT;

V\_NET\_TYPE := P\_TYPE;

V\_NET\_AMOUNT := V\_PREV\_AMOUNT + P\_AMOUNT;

**-- Update the record in Card Master table**

UPDATE CARD\_MASTER

SET PREV\_TPYE = V\_PREV\_TYPE,

PREV\_AMOUNT = V\_PREV\_AMOUNT,

CURR\_TYPE = V\_CURRENT\_TYPE,

CURR\_AMOUNT = V\_CURRENT\_AMOUNT,

NET\_TYPE = V\_NET\_TYPE,

NET\_AMOUNT = V\_NET\_AMOUNT,

LAST\_TRANS\_DATE = P\_TRANS\_DATE

WHERE CARD\_ID = P\_CARD\_ID;

END IF; **-- Insert the transaction details into Card Details table**

INSERT INTO CARD\_DETAILS (CARD\_ID, TYPE, AMOUNT, TRANS\_DATE)

VALUES (P\_CARD\_ID, P\_TYPE, P\_AMOUNT, P\_TRANS\_DATE);

END;

**Executing a Procedure**

EXEC card\_proc('C1', 'John', 'D', 1000, TO\_DATE('10/10/2021', 'DD/MM/YYYY'));

EXEC card\_proc('C2', 'Raj', 'D', 500, TO\_DATE('12/10/2021', 'DD/MM/YYYY'));

EXEC card\_proc('C3', 'Williams', 'C', 100, TO\_DATE('15/10/2021', 'DD/MM/YYYY'));

**Result:**

